United International University (UIU)
Dept. of Computer Science & Engineering (CSE)
Final Exam :: Summer 2022

Course Code: CSE 1115 Course Title: Object Oriented Programming

Total Marks: **40** Time: **2 hours**

READ THIS CAREFULLY: Any examinee found adopting unfair means will be expelled from the trimester / program as per UIU disciplinary rules

Answer all the four questions from [Question 1 to 4]. There are two questions named Question 4. You need to answer one of them.

Question 1 [1 + 4 + 5]

A. Write difference between abstract class and interface. What will happen if we want to change the value of a field in interface after initialization?

B. Write two interfaces. Interface I1 contains a method called "**methodA**". Interface I2 contains a method called "**methodB**". Now write a class "**InterfaceTest**" and implement the Interface B in such a way that both methods "**methodA**" and "**methodB**" are implemented in the "**InterfaceTest**" class.

C. Find the errors (if there is any) in both of the two code blocks below:

```
public class StaticBlock {
  public static int a = 5, b;
  public int c;
  static {
    b = c * 4;
  }
  static {
    c = 5;
  }
  public static void main(String[] args) {
    new StaticBlock();
  }
}
```

```
class point{
  public int x, y, z;
  public point(int x, int y, int z){
    this.x = x;
    this.y = y;
    this.z = z;
  }
}
class FinalCheck {
  public static final point p = new point(1, 2, 3);
  public FinalExam(){
    System.out.println("Final is both exam and keyword");
  }
  public static void check(){
    p.x = 3;
    p.y = 4;
    p = new point(2, 3, 4);
  }
  public static void main(String[] args) {
    check();
  }
}
```

Question 2 [4 + 6]

A. Write a program in Java to count the number of lines from a file named "in.txt" and copy all the contents from the file to another file named "out.txt".

B. Write **added/updated codes only** to complete the following **three** tasks:

```
class InvalidUserException [extend the Exception class]
{
// Write your answer for (i). here.
}
```

```
class ExceptionDemo {
  void UserCheck(int age, int work_experience ) throws
InvalidUserException{
    // Write your answer for (ii). here.
  }
}
```

```
public static void main(String args[])  {
            ExceptionDemo obj = new ExceptionDemo();
            Scanner input = new Scanner(System.in);
            int age = input.nextInt();
            int work_experience = input.nextInt();
           // Write your answer for (iii). here.
            obj.UserCheck(age, work_experience);
  }
}
```

i. Create a user-defined exception class named InvalidUserException that extends the Exception class and prints the messages provided below using the super constructor.

ii. Provide logical implementation so that if age is below 50 then throw an exception of InvalidUserException class with the message "Ineligible for Elderly pension because of age", if work_experience is below 20 then throw an exception of InvalidUserException class with the message "Ineligible for Elderly pension because of work experience".

iii. Invoke the UserCheck() method using proper try-catch block. You catch block should print a message "Caught the exception" and print the correct message from (ii).

Question 3 [6 + 4]

Complete the following **two** tasks:

a) Write the code to get a Java GUI application like Image 3(a) below that has the functionality of converting Foot to Inch after pressing the Convert button. Assume all the packages are imported. User can input decimal numbers in the input fields (e.g. 10, 5.5 etc). *Formula: 1 foot = 12 Inch*

b) In the previous code of Question (a), add another button ConvertToFoot beside ConvertToInch like the figure in Image 3(b). Now, if Convert to Inch is pressed, the value of foot (given in the Foot TextField) will be converted to inch as previous and if Convert to Foot is pressed, the value of inch (given in the Inch TextField) will be converted to foot (The result will be shown in the Foot TextField). **Just write the updated/added codes over Question (a)**

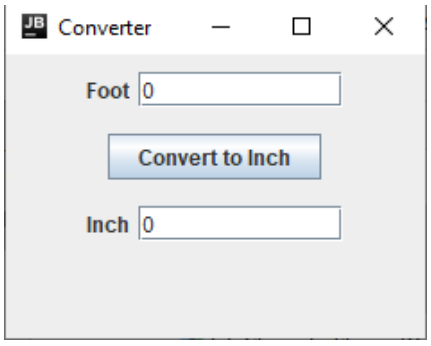

Image 1: GUI for question 3(a)

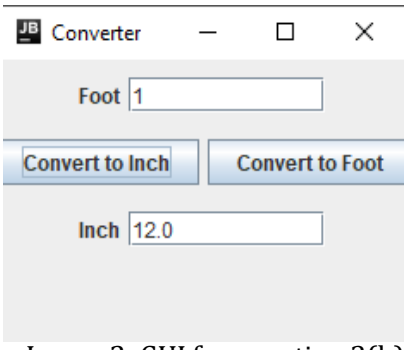

Image 2: GUI for question 3(b)

Question 4 [3 + 3 + 4]

Consider the following codes and complete the three tasks below:

i. Override the **equals** method in the Student class so that two students are considered equal if their name is matched.

ii. Call the **contains** method of the alist object in **Line No 26** to check whether the Student object: query exists in it or not.

iii. Sort the alist in descending order of age of the Students **must using an anonymous inner class** of the Comparator (You must use the **sort** method of the Collections class)

N.B. Just write the added/updated codes only.

```
1  import java.util.*;
2
3
4  class Student{
5      public String name;
6      public int age;
7      public double cgpa;
8      public Student(String name, int age, double cgpa) {
9          this.name = name;
10         this.age = age;
11         this.cgpa = cgpa;
12     }
13
14     // Task-(a): Override the equals method that returns true if name of two
15 students are equal
16 }
17
18 public class ArrayListDemo {
19     public static void main(String[] args) {
20         ArrayList<Student> alist = new ArrayList<>();
21         alist.add(new Student("Sonet", 15, 3.8));
22         alist.add(new Student("Zhang", 17, 3.9));
23         alist.add(new Student("Buffon", 20, 3.6));
24
25         Student query = new Student("Zhang", 0, 0); // this should exist in alist
26         // Task-(b): Properly call the contains method of the alist to find
27 whether the object query exists in it or not
28
29         // Task-(c): Write the proper custom comparator and sort the alist using
30 sort method of the Collections class
31     }
32 }
```

Or,

Question 4 [4 + 3 + 3]

Create a thread class named **SumThread** that takes three parameters in the constructor: an **array**, an index **left**, another index **right**. The thread computes the sum of the passed array from index l to r **inclusive when it runs**. Consider the following main method of the Main class provided below and the array a. *You can assume the input n, will be multiple of 3*.

```
public class Main {
    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter num of elements: ");
        int n = scanner.nextInt();

        int[] a = new int[n];
        int total_sum = 0;

        for (int i = 0; i < a.length; i++) {
            a[i] = scanner.nextInt();
        }

        // Write your codes here
        // .....................

        System.out.println("Total Sum: " + total_sum);
    }
}
```

Now complete the following three tasks:

a. Create the **SumThread** class by **implementing the Runnable interface** and define the constructor and other required methods. Pass the values of *left* & *right* properly to every thread constructor so that every object can compute the sum of **one-third non-overlapping** partitions.

b. Start three threads of the **SumThread** object. **The threads must run and compute the sum concurrently**.

c. After that, set the **accurate total sum from these three threads** into the ***total_sum*** variable in the main method of the Main class.
You must handle relevant exceptions. You don't need to rewrite the whole Main class. Just write the added codes.